Thiết kế mạch số với HDL

cuu duong than cong . com

Chương 4: Thiết kế luận lý với Verilog

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Nội dung chính

- Giới thiệu về HDLs và verilog
- Mô hình cấu trúc cho mạch luận lý tổ hợp
- Mô phỏng luận lý, kiểm chứng thiết kế và phương pháp luận kiểm tra
- Thời gian trễ truyền lan
- Mô hình bảng sự thật cho mạch luận lý tổ hợp và tuần tự với Verilog

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Nội dung chính

- **Giới thiệu về HDLs và verilog**
- Mô hình cấu trúc cho mạch luận lý tổ hợp
- Mô phỏng luận lý, kiểm chứng thiết kế và phương pháp luận kiểm tra
- Thời gian trễ truyền lan
- Mô hình bảng sự thật cho mạch luận lý tổ hợp và tuần tự với Verilog

cuu duong than cong . com

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Computer Engineering 2008

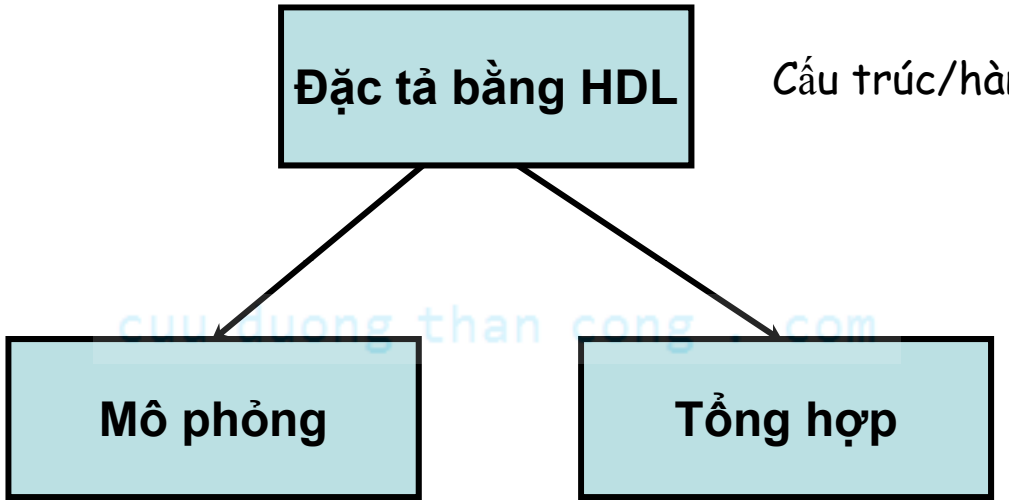
Giới thiệu HDLs

- HDLs (Hardware Description Languages)
  - Không là một ngôn ngữ lập trình
  - Tựa C
  - Thêm những chức năng mô hình hóa, mô phỏng chức năng
  - Verilog vs. VHDL
- Các bước thiết kế bằng HDL
  - Mô tả mạch từ khóa
  - Biên dịch để kiểm tra cú pháp (syntax)
  - Mô phỏng để kiểm tra chức năng của mạch

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Phương pháp luận thiết kế HDL



Cấu trúc/hành vi của mạch

Kiểm tra: thiết kế đã đúng yêu cầu chưa?
Chức năng: Hành vi I/O
Mức thanh ghil (Kiến trúcl)
Mức luận lý (Cổng)
Mức transistor (Điển tử)
Timing: Waveform Behavior

Ánh xạ đặc tả thành các hiện thực

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Mô hình cấu trúc và mô hình hành vi trong HDLs

- *Cấu trúc (Structural)* chỉ tra cấu trúc phần cứng thật sự của mạch
  - Mức trừu tượng thấp
    - Các cổng cơ bản (ví dụ and, or, not)
    - Cấu trúc phân cấp thông qua các module
  - Tương tự lập trình hợp ngữ
- *Hành vi (Behavioral)* chỉ ra hoạt động của mạch trên các bits
  - Mức trừu tượng cao hơn
    - Biểu diễn bằng các biểu thức (ví dụ out = (a & b) | c)
- Không phải tất cả các đặc tả hành vi đều tổng hợp được
  - Không sử dụng: + - * / % > >= < <= >> <<

Computer Engineering 2008
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Những nguy hiểm trong thiết kế Verilog

- Chương trình tuần tự, bộ tổng hợp có thể sẽ phải thêm nhiều chi tiết phần cứng
  - Cần một bộ priority encoder
- Nếu chương trình song song, có thể có những trạng thái không xác định
  - Nhiều khối "always", khối nào thực thi trước?
- Tạo ra nhiều trạng thái không dự dịnh trước

```
if (x == 1) out = 0;
if (y == 1) out = 1;    // else out retains previous state?
   R-S latch!
```

- Không tính trước được số phần tử phần cứng
  - x = x + 1 có thể cần RẤT NHIỀU phần tử phần cứng

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Lịch sử phát triển HDLs

- ISP (circa 1977) – dự án nghiên cứu CMU (Carnegie Mellon University)
  - Mô phỏng nhưng không tổng hợp
- Abel (circa 1983) – được phát triển bởi Data-I/O
  - Mục tiêu dùng cho các thiết bị luận lý khả lập trình
  - Không tốt cho máy trạng thái
- Verilog (circa 1985) – phát triển bởi Gateway (now Cadence)
  - Đặc tả được đưa ra từ 1985
  - Ban đầu được phát triển cho mô phỏng, tương tự C và Pascal
  - Hiệu quả và dễ viết
  - Berkeley phát triển công cụ tổng hợp vào thập niên 80
  - Được IEEE chuẩn hóa
    - Verilog standardized (Verilog-1995 standard)
    - Verilog-2001 standard
- VHDL (circa 1987) - DoD sponsored standard
  - Dưa trên VHSIC phát triển bởi DARPA
  - Tương tự như Ada (Nhấn mạnh vào tái sử dụng và bảo trì)
  - Ngữ nghĩa mô phỏng rõ ràng
  - Rất tổng quát nhưng dài dòng
  - Được IEEE chuẩn hóa
    - VHDL standardized (’87 and ’93)
  - Cấu trúc nghiêm ngặt

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Verilog HDL

- Verilog là một ngôn ngữ lớn
  - Có nhiều tính năng cho tổng hợp và mô phỏng phần cứng
  - Có thể biểu diễn những đặc trưng mức thấp
    - Transistor
  - Có thể hoạt động như ngôn ngữ lập trình
    - Cấu trúc lặp
    - Cấu trúc điều khiển….
- Các công cụ mô phỏng chấp nhận toàn bộ khái niệm của Verilog
- Các công cụ tổng hợp chỉ chấp nhận một phần các khái niệm của Verilog
- Chỉ tập trung nghiên cứu một phần
  - Sử dụng ở một mức thích hợp
  - Tập trung trên những cấu trúc tổng hợp được
  - Tập trung tránh những cấu trúc gây lỗi khi tổng hợp

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Nội dung chính

- Giới thiệu về HDLs và verilog
- **Mô hình cấu trúc cho mạch luận lý tổ hợp**
- Mô phỏng luận lý, kiểm chứng thiết kế và phương pháp luận kiểm tra
- Thời gian trễ truyền lan
- Mô hình bảng sự thật cho mạch luận lý tổ hợp và tuần tự với Verilog

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Mô hình mạch tổ hợp

- Một mô hình Verilog của một mạch tóm tắt các mô tả chức năng bằng góc nhìn cấu trúc hay hành vi trên những mối quan hệ ngõ vào-ngõ ra
- Một mô hình cấu trúc là một cấu trúc kết nối (netlist) chứa
  - Các cổng
  - Các khối chức năng
- Một mô hình hành vi là
  - Các biểu thức Boolean đơn giản
  - Mô hình chuyển đổi mức thanh ghi (Register Transfer Level – RTL)
  - Một giải thuật

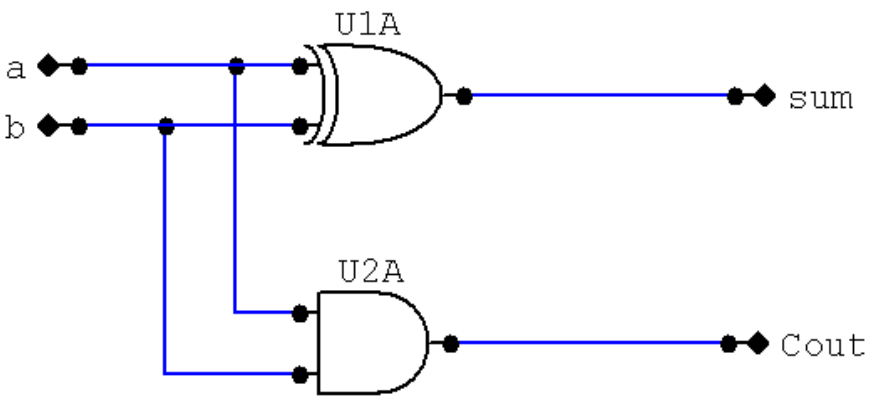
cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Mô hình cấu trúc mạch tổ hợp

- Thiết kế cấu trúc tương tự như tạo ra một sơ đồ (schematic)
- Schematic
  - Hình biểu diễn cổng logic,
  - Ngõ vào ra,
  - Các đường kết nối giữa các cổng.
- Mô hình cấu trúc HDL
  - Danh sách các cổng cơ bản và kết nối giữa chúng
  - Các phát biểu chỉ ra ngõ vào-ra



cuu duong than cong . com

D:/Working/Verilog/halfadder

File Edit View Tools

ln #

```
1   module half_adder (sum, Cout, a, b);
2       input a, b;
3       output sum, Cout;
4
5       xor (sum, a, b);
6       and (Cout, a, b);
7
8   endmodule
```

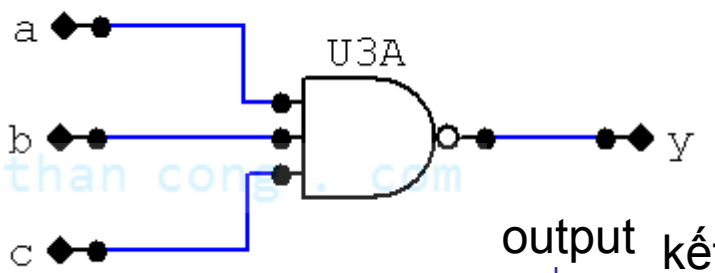
cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Verilog primitives

- Primitives là các đối tượng cơ bản có thể được sử dụng trong thiết kế
- 26 đối tượng chức năng được định nghĩa trước

n-input	n-output 3-states
and	buf
nand	not
or	bufif0
nor	bufif1
xor	notif0
xnor	notif1



output

kết thúc phát biểu

nand (y, a, b, c);

input

keyword name

nand N1(y, a, b, c);

instance name (optional)

Ngõ ra là phần tử đầu tiên trong danh sách các ngõ vào-ra

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Mô hình cấu trúc trong Verilog

- ***Module***
  - Tên module và đi theo sau là danh sách các ngõ vào-ra (port)
  - Danh sách đặc tả loại port (input/output)
  - Danh sách các dây nối, các biến sử dụng bên trong module (optional)
  - Danh sách các kết nối giữa các cổng và các module khác bên trong
  - endmodule

```
module module_name (port_list);
//Declarations:
    reg, wire, parameter,
    input, output, inout,
    function, task, …
//Statements:
    Initial statement
    Always statement
    Module instantiation
    Gate instantiation
    UDP instantiation
    Continuous assignment
endmodule
```

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Ví dụ

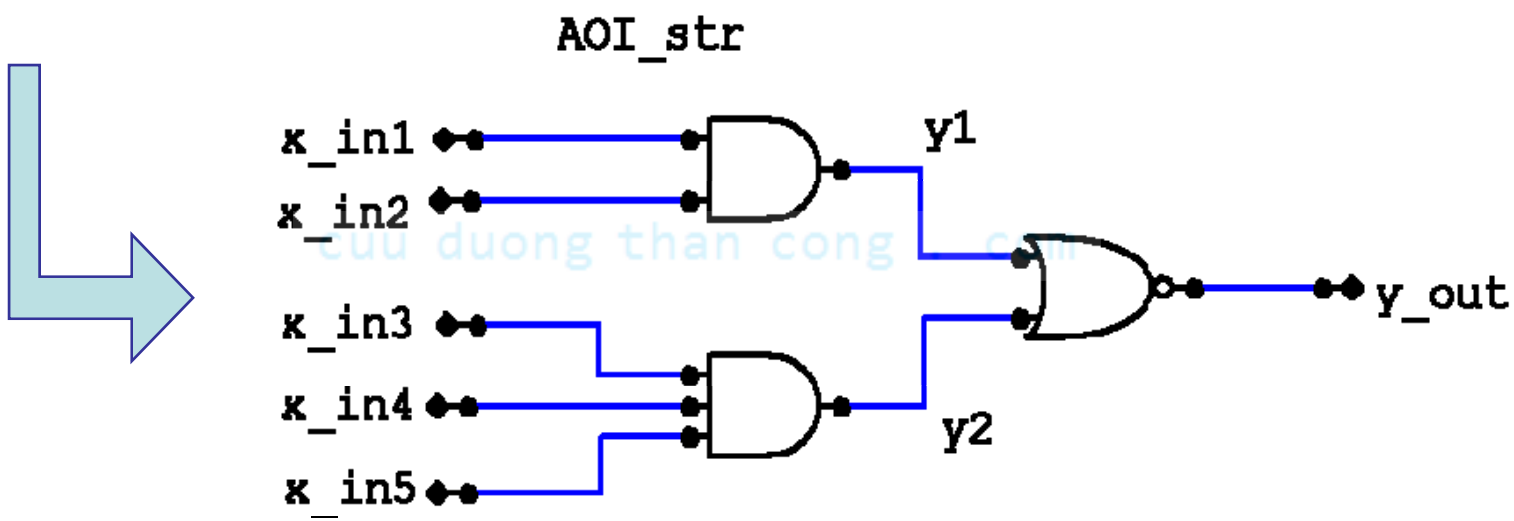
```
ln #
1   module AOI_str (y_out, x_in1, x_in2, x_in3,
2                   x_in4, x_in4);
3       output  y_out;
4       input x_in1, x_in1, x_in2, x_in3,
5             x_in4, x_in4;
6
7       wire y1, y2;
8
9       nor (y_out, y1, y2);
10      and (y1, x_in1, x_in2);
11      and (y2, x_in3, x_in4, x_in5);
12  endmodule
```

Module ports

port modes

Internal wires

Instantiated primitives



cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Module ports

- Giao tiếp với "môi trường" bên ngoài
- Kiểu của port quyết định chiều truyền dữ liệu
  - Một chiều (Unidirectional)
    - input
    - output
  - Hai chiều (Bidirectional)
    - inout
- Kiểu của module port phải được khai báo tường minh và không cần theo thứ tự xuất hiện trong port list

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Quy tắt trong Verilog

- Phân biệt chữ hoa thường (Case sensitive)
- Identifier: a-z, A-Z, 0-9, '_' và '$'
- Tên biến không được bắt đầu bằng '$' hay ký số và có thể tối đa là 1024 ký tự
- Một phát biểu được kết thúc bằng ';'
- Chú thích
  - '//' một dòng chú thích
  - /*…*/ chú thích nhiều dòng
- Các identifiers có phạm vi nhất định
- Có thể viết các phát biểu trên một dòng hay nhiều dòng

Computer Engineering 2008
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Thiết kế từ trên xuống (top-down)

- Hệ thống phức tạp được phân chia thành những đơn vị chức năng nhỏ hơn
  - Dễ thiết kế
  - Dễ kiểm tra
- Các module lồng nhau trong Verilog hỗ trợ thiết kế từ trên xuống
- Module tham khảo đến module khác được gọi là module "cha", module được module khác tham khảo đến gọi là module "con"
- Độ sâu của các module lồng nhau không giới hạn
- Mỗi module con phải có tên duy nhất trong phạm vi module cha (trừ các primitives)

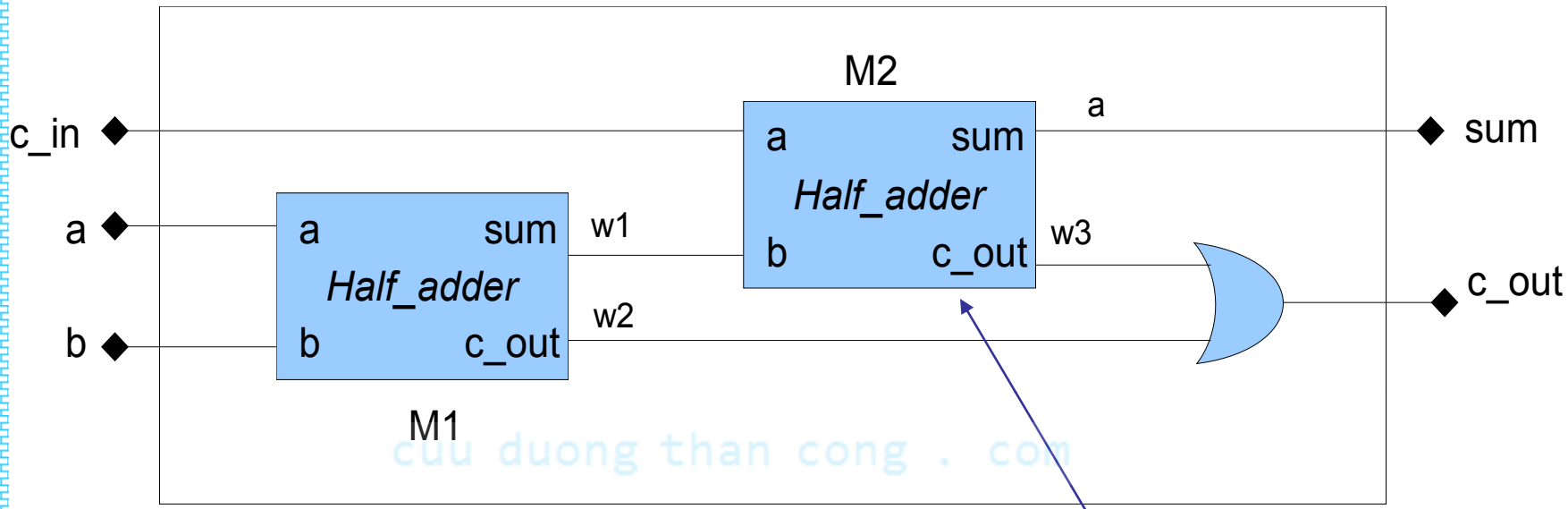
cuu duong than cong . com

cuu duong than cong . com

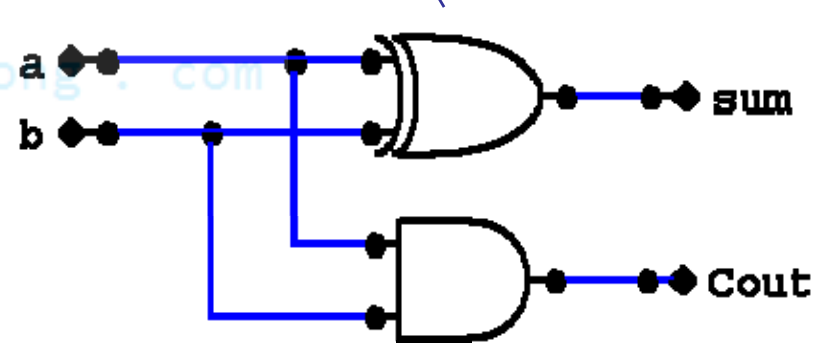
Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Binary full adder




```
ln #
1   `include "halfadder.v"
2
3   module full_adder (sum, c_out, a, b, c_in);
4       input a, b, c_in;
5       output sum, c_out;
6       wire w1, w2, w3;
7
8       half_adder M1 (w1, w2, a, b);
9       half_adder M2 (sum, w3, c_in, w1);
10      or (c_out, w2, w3);
11  endmodule
```



Computer Engineering 2008
cuu duong than cong . com
CuuDuongThanCong.com
https://fb.com/tailieudientucntt

Thiết kế phân cấp và tổ chức mã nguồn

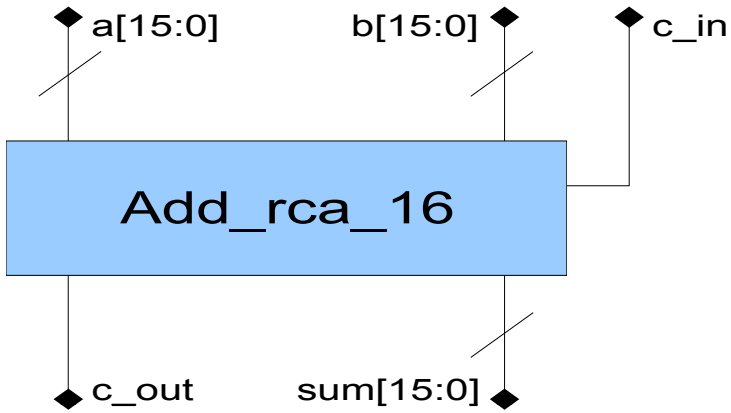
- Top-level module là module ở cấp cao nhất
- Module ở mức thấp nhất
  - Chứa các primitives
  - Các module không phân chia nhỏ hơn
- Tất cả các module được đặt trong một hay nhiều tập tin khác nhau
- Công cụ mô phỏng tích hợp các module từ các tập tin

cuu duong than cong . com

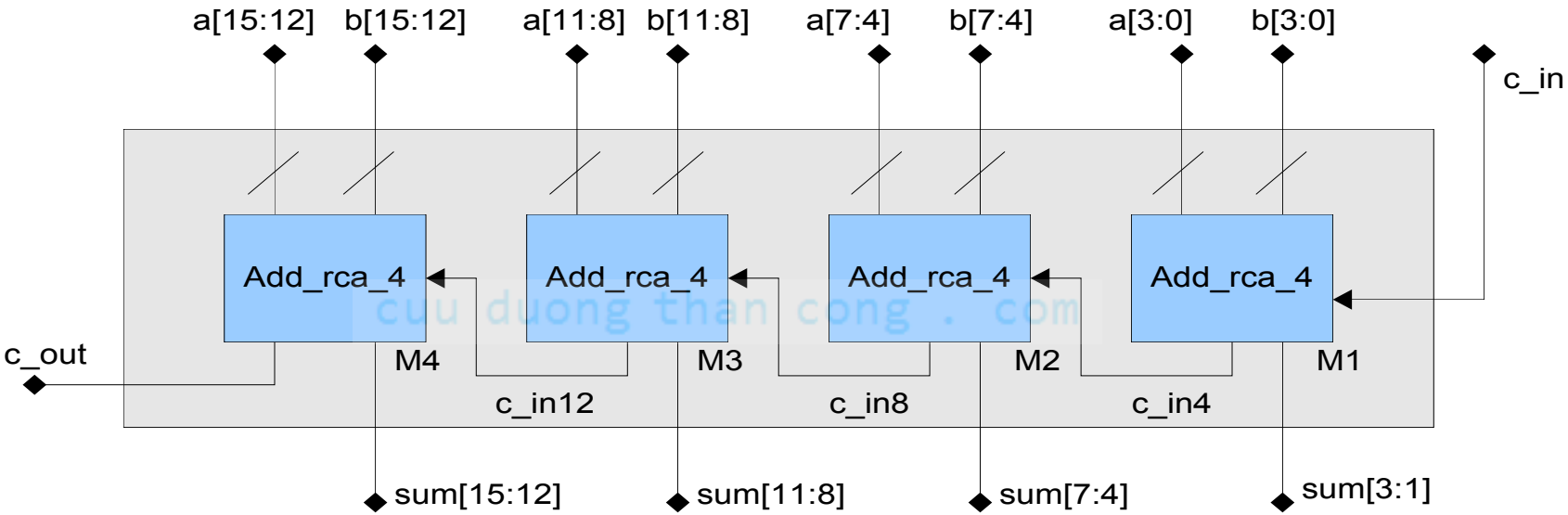
Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Mạch cộng 16-bit ripple carry



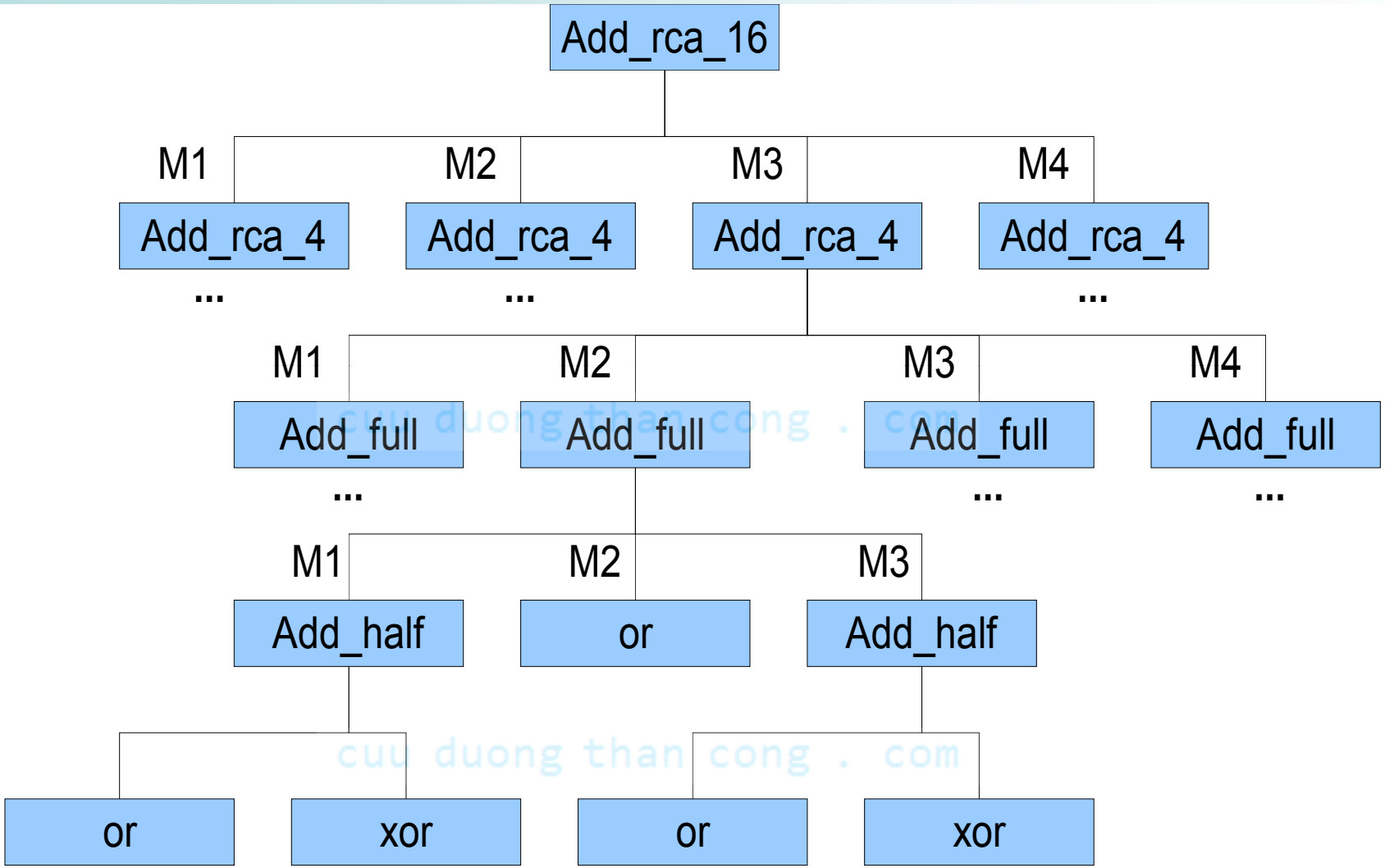
cuu duong than cong . com



cuu duong than cong . com

Computer Engineering 2008
CuuDuongThanCong.com
https://fb.com/tailieudientucntt

Cây phân cấp mạch cộng 16-bit ripple carry



Cây phân cấp thiết kế mạch cộng 16 bit ripple carry

Computer Engineering 2008
cuu duong than cong . com
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hiện thực mạch cộng 16-bit ripple carry

D:/Working/Verilog/add_rca_16.v

File Edit View Tools

```
module Add_rca_16 (sum, c_out, a, b, c_in);
    output[15:0] sum;
    output c_out;
    input[15:0] a, b;
    input c_in;
    wire c_in4, c_in8, c_in12;

    Add_rca_4 M1 (sum[3:0], c_in4, a[3:0], b[3:0], c_in);
    Add_rca_4 M2 (sum[7:4], c_in8, a[7:4], b[7:4], c_in4);
    Add_rca_4 M3 (sum[11:8], c_in12, a[11:8], b[11:8], c_in8);
    Add_rca_4 M4 (sum[15:12], c_in4, a[15:12], b[15:12], c_in12);
endmodule

module Add_rca_4 (sum, c_out, a, b, c_in);
    output[3:0] sum;
    output c_out;
    input[3:0] a, b;
    input c_in;
    wire c_in2, c_in3, c_in4;

    full_adder M1 (sum[0], c_in2, a[0], b[0], c_in);
    full_adder M2 (sum[1], c_in3, a[0], b[0], c_in2);
    full_adder M3 (sum[2], c_in4, a[0], b[0], c_in3);
    full_adder M4 (sum[3], c_out, a[0], b[0], c_in4);
endmodule
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Vectors trong Verilog

- Một vector được biểu diễn bằng ngoặc vuông chứa dãy liên tiếp các bit
  - sum[3:0] vector sum kích thước 4 bit
- Bit trái nhất là MSB
- Bit phải nhất là LSB
- Có thể truy xuất từng bit hay từng dãy bit trong vector
  - sum[1] bit thứ 2 từ phải sang của sum
  - sum[2:1] bit thứ 2 và 3 từ phải sang của sum
- sum[4] giá trị x (không xác định)
- Có thể gán, so sánh 2 vector với nhau

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Cấu trúc liên kết (connectivity)

- Wire
  - Thiết lập các liên kết giữa các đối tượng thiết kế
  - Giá trị được quyết định trong quá trình mô phỏng bởi cái mà nó được nối vào
- Kiểu wire
  - Khai báo bằng từ khóa wire
    - wire y_out
  - Các biến sử dụng không khai báo
- Các ngõ vào và ra mặc định là kiểu wire (trừ khi được khai báo kiểu khác)
- Kết nối giữa port hình thức và port thực tế
  - Theo thứ tự trong danh sách các port
  - *.tên_hình_thức(tên_thực_tế)*
    - half_adder (.b(b), .Cout(w2), .a(a), .sum(w1));

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Nội dung chính

- Giới thiệu về HDLs và verilog
- Mô hình cấu trúc cho mạch luận lý tổ hợp
- **Mô phỏng luận lý, kiểm chứng thiết kế và phương pháp luận kiểm tra**
- Thời gian trễ truyền lan
- Mô hình bảng sự thật cho mạch luận lý tổ hợp và tuần tự với Verilog

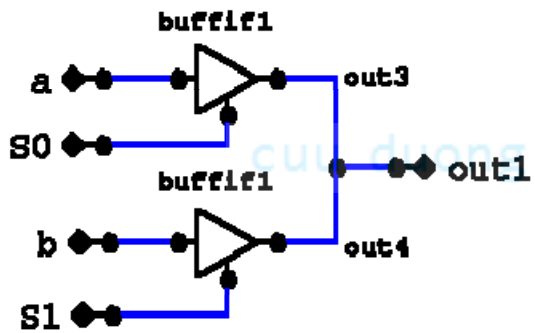
cuu duong than cong . com

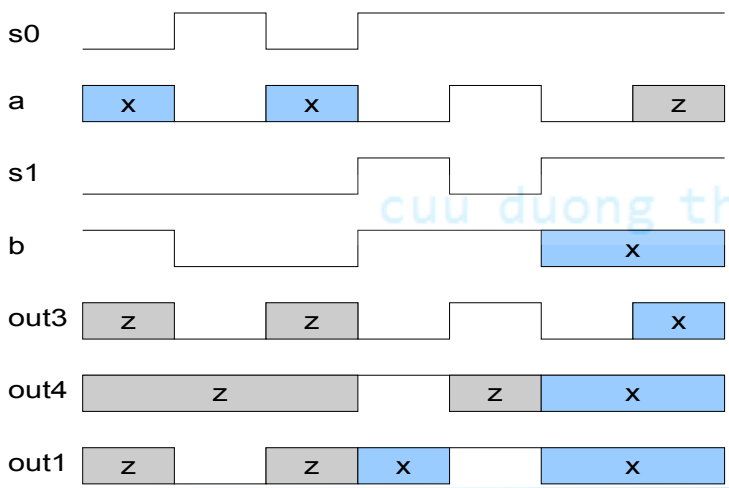
cuu duong than cong . com

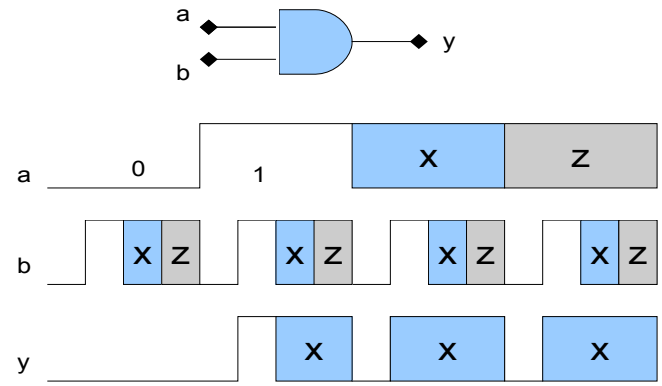
CuuDuongThanCong.com https://fb.com/tailieudientucntt

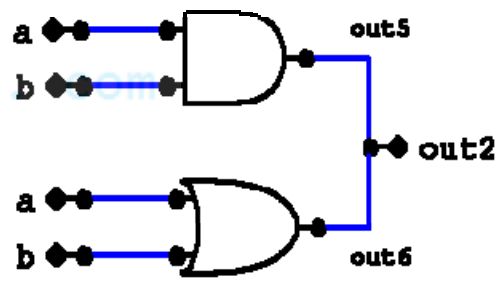
Các giá trị luận lý trong Verilog

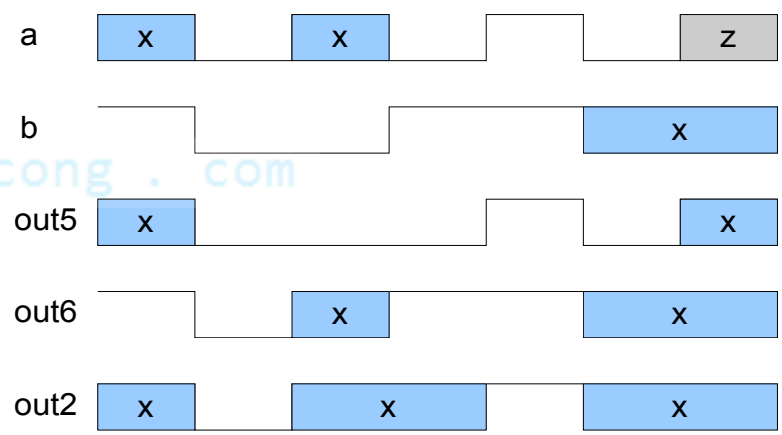
- Verilog sử dụng 4 giá trị luận lý
  - 1 True
  - 0 False
  - x Không xác định
  - z tổng trở cao











cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

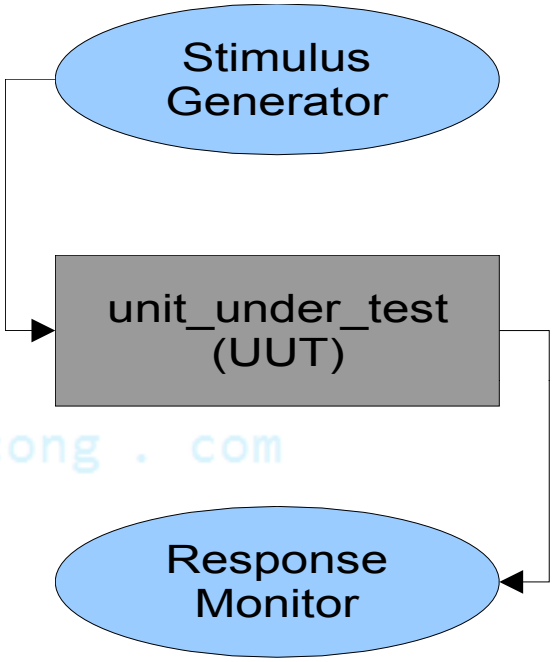
Phương pháp luận kiểm tra

- Kiểm tra mạch thực hiện đúng chức năng
  - Kiểm tra ngẫu nhiên phức tạp và không chính xác
  - Cần lập kế hoạch kiểm tra tỉ mỉ
- Kiểm tra mạch lớn
  - Kiểm tra tất cả các trường hợp
    - Mạch cộng 16 bit cần kiểm tra 2^{23} trường hợp
    - Kiểm tra phân cấp
      - half_adder
      - full_adder
      - Add_rca_4 cần kiểm tra 2^9 trường hợp
      - Chọn một số trường hợp để kiểm tra kết nối của các Add_rca_4 trong Add_rca_16
- Kiểm tra theo chiều ngược so với cây phân cấp thiết kế

Computer Engineering 2008
cuu duong than cong . com
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Mô phỏng luận lý

- Xây dựng các testbench đưa vào mạch và hiển thị dạng sóng của kết quả
- Bộ mô phỏng
  - Kiểm tra mã nguồn
  - Báo lỗi
  - Mô phỏng hành vi của mạch thông qua các tín hiệu vào trong testbench



Người sử dụng hay phần mềm

Không có lỗi cú pháp $\not\Rightarrow$ Mạch thực thi đúng kết quả

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Sinh tín hiệu để kiểm tra

- Một hành vi (behavior)
  - Tập hợp các phát biểu được thi trong quá trình mô phỏng (các phát biểu thủ tục)
  - Được gán giá trị mô phỏng giống như là được điều khiển bởi phần cứng
- *Initial* khai báo hành vi một lượt (single-pass)
- *begin … end*
  - Chứa danh sách các phát biểu của hành vi
  - Thời gian thực thi các phát biểu thủ tục tùy thuộc vào thứ tự và thời gian trễ truyền
  - Các phát biểu được thực thi từ trên xuống, từ trái sang phải
- *# <integer> <statement>*
  - Điều khiển trễ truyền
  - Các phát biểu phía sau phải đợi

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Khuôn mẫu cho testbench

```
module t_module ();
    reg …;


    wire …;
    parameter time_out = 100;
    UUT_name U1 (port_list);
    initial $monitor();

    initial #time_out $finish
    initial
        begin … end

endmodule
```

Tên module UUT

Khai báo kiểu thanh ghi để chứa giá trị cho các biến ngõ vào của UUT

Khai báo kiểu wire cho các ngõ ra của UUT

Khai báo các tham số

Tín hiệu được xuất ra dạng text

Kết thúc mô phỏng sau thời gian time_out

Xây dựng các tín hiệu ngõ vào cho U1

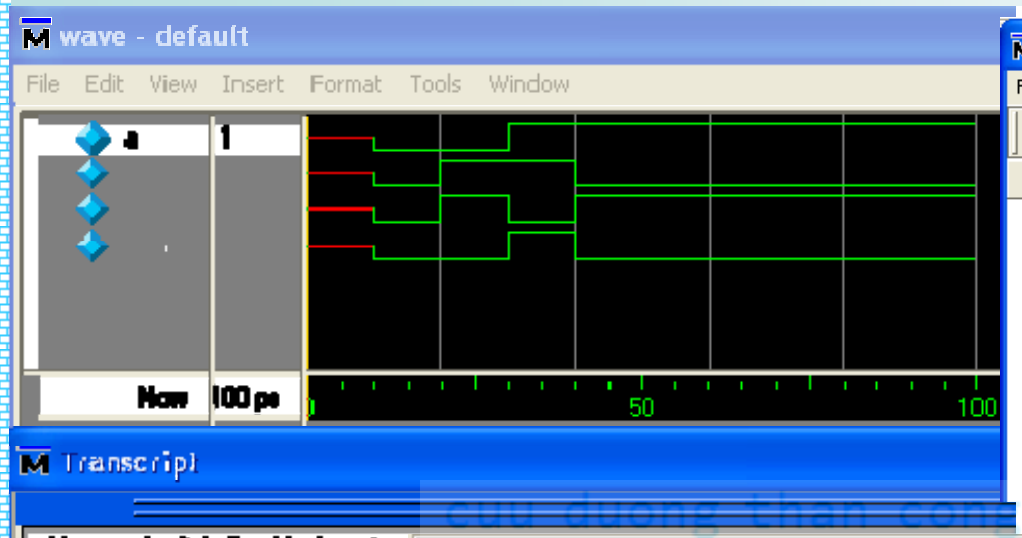
cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Ví dụ



D:/Working/Verilog/halfadder

File Edit View Tools

```
1   module half_adder (sum, Cout, a, b);
2       input a, b;
3       output sum, Cout;
4
5       xor (sum, a, b);
6       and (Cout, a, b);
7
8   endmodule
```

```
add wave dirc/t_half_adder/c_out
VSIM 9> run
# a = x, b = x, sum = x, c_out = x
# a = 0, b = 0, sum = 0, c_out = 0
# a = 0, b = 1, sum = 1, c_out = 0
# a = 1, b = 1, sum = 0, c_out = 1
# a = 1, b = 0, sum = 1, c_out = 0
```

D:/Working/Verilog/test_half_adder.v

```
1   module t_half_adder();
2       wire sum, c_out;
3       reg a, b;
4       parameter time_out = 100;
5       half_adder M1(sum, c_out, a, b);
6       initial
7           $monitor("a = %b, b = %b, sum = %b, c_out = %b",
8                    a, b, sum, c_out );
9       initial #time_out $finish;
10      initial begin
11          #10 a = 0; b = 0;
12          #10 b = 1;
13          #10 a = 1;
14          #10 b = 0;
15      end
16  endmodule
```

cuu duong than cong . com

CuuDuongThanCong.com
https://fb.com/tailieudientucntt

Kích thước số

- Chỉ ra số lượng bit dùng để lưu trữ số
- *<num_of_bit> ‘ <base> <value>*
  - 8’b1000_0001
  - 3’d2987
  - 16’h24ce
  - 3’o7
- Những số không khai báo kích thước được hiểu ở dạng integer (thông thường 32 bits)
- Mặc định các số ở dạng decimal
- Không phân biệt hoa thường trong khai báo số
- Dấu “_” được bỏ qua

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Nội dung chính

- Giới thiệu về HDLs và verilog
- Mô hình cấu trúc cho mạch luận lý tổ hợp
- Mô phỏng luận lý, kiểm chứng thiết kế và phương pháp luận kiểm tra
- **Thời gian trễ truyền lan**
- Mô hình bảng sự thật cho mạch luận lý tổ hợp và tuần tự với Verilog

cuu duong than cong . com

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

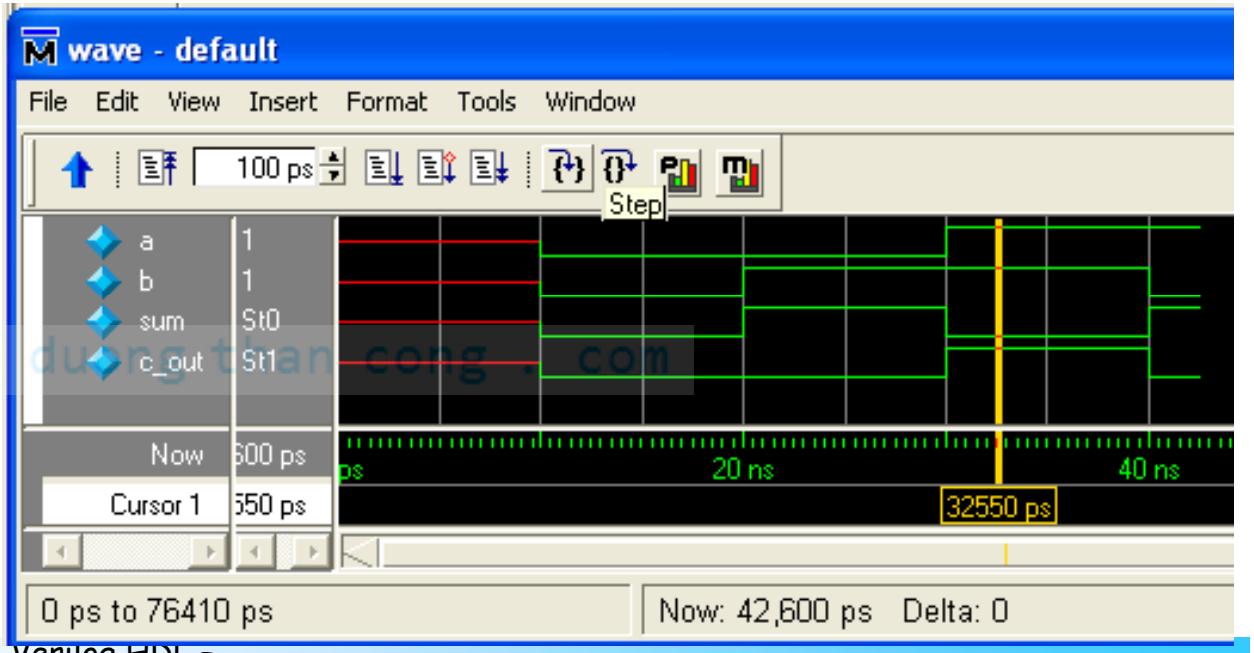
Thời gian trễ truyền

- Ngõ vào thay đổi ngõ ra không thay đổi ngay lập tức
- Các phần tử cơ bản của verilog có thời gian trễ là 0
- Các vi mạch thực tế được sản xuất dựa trên các thư viện chuẩn được định nghĩa trước
- Người thiết kế chỉ quan tâm đến tính đúng đắn của mạch
- Sử dụng các công cụ tổng hợp để hiện thực các thiết kế thỏa mãn các ràng buộc thời gian
- *'timescale <time_unit base>/<precision base>*
  - Chỉ thị biên dịch
  - Chỉ ra đơn vị thời gian và độ chính xác thời gian trễ
  - Phải được khai báo trước khi các module

Computer Engineering 2008
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Ví dụ `timescale

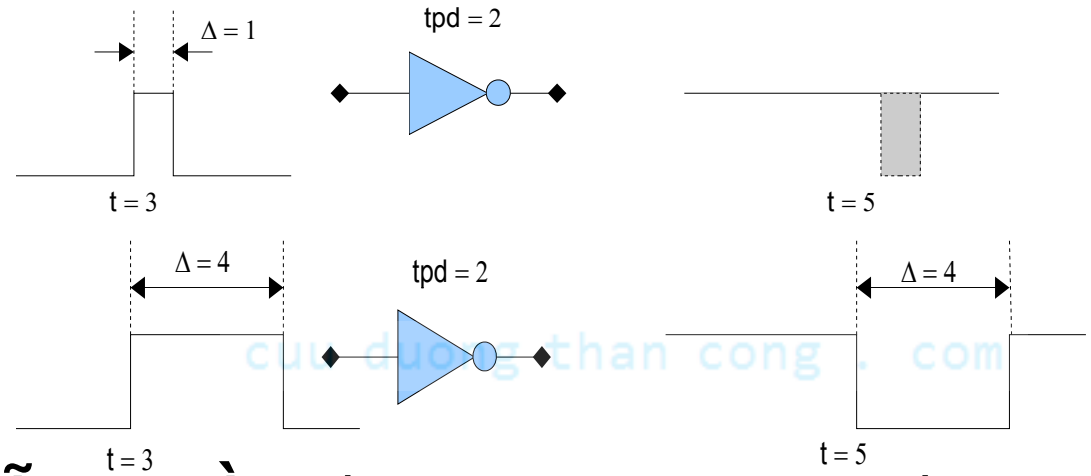
```
1   `timescale 1ns/10ps
2   module t_half_adder();
3       wire sum, c_out;
4       reg a, b;
5       parameter time_out = 130;
6       half_adder M1(sum, c_out, a, b);
7       initial
8           $monitor("a = %b, b = %b, sum = %b, c_out = %b",
9                    a, b, sum, c_out );
10      initial #time_out $finish;
11      initial begin
12          #10 a = 0; b = 0;
13          #10 b = 1;
14          #10 a = 1;
15          #10 b = 0;
16      end
17  endmodule
```



CuuDuongThanCong.com https://fb.com/tailieudientucntt

Các loại trễ lan truyền

- Trễ quán tính (inertial delay)



- Trễ truyền (transport delay)
  - Gây ra do các dây nối
  - 0.033ns/1cm
  - Có thể bỏ qua
  - *wire #2 long_wire*

cuu duong than cong . com

cuu duong than cong . com

Computer Engineering 2008

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Nội dung chính

- Giới thiệu về HDLs và verilog
- Mô hình cấu trúc cho mạch luận lý tổ hợp
- Mô phỏng luận lý, kiểm chứng thiết kế và phương pháp luận kiểm tra
- Thời gian trễ truyền lan
- **Mô hình bảng sự thật cho mạch luận lý tổ hợp và tuần tự với Verilog**

cuu duong than cong . com

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Bảng sự thật trong Verilog

- *table*
- Ngõ ra phải có kiểu vô hướng (scalar)
- Dùng kí hiệu '?' thay cho 0, 1, x
- Thứ tự các cột trong <input_list> tương ứng với thứ tự trong khai báo input của module
- Mạch tổ hợp
  - *<input_list>:<output>*
- Mạch tuần tự
  - *<input_list>:<state>:<output/next_state>*
  - Ngõ ra phải được khai báo kiểu thanh ghi
  - Dùng kí hiệu '-' biểu diễn ngõ ra không thay đổi

Computer Engineering 2008
CuuDuongThanCong.com https://fb.com/tailieudientucntt